

**Centro de Enseñanza Técnica Industrial**

**Desarrollo de Software**

**Analizador sintáctico**

**Jesús Alberto Aréchiga Carrillo**

**22310439 6N**

**Profesor**

**Darío Yerenas**

**Junio 2025**

**Guadalajara, Jalisco**

Archivo de flex:

%%

"SetUnion"|"ClearSet"|"PrintSet"|"Showsets"|"Delete"|"Union"|"intersection"|"Sets"|"Set" {

printf("|CMD, %s|", yytext);

}

":=" { printf("|ASSIGN, %s|", yytext); }

"{" { printf("|LBRACE, %s|", yytext); }

"}" { printf("|RBRACE, %s|", yytext); }

"," { printf("|COMMA, %s|", yytext); }

";" { printf("|SEMICOLON, %s|", yytext); }

[A-Za-z][A-Za-z0-9]\* {

printf("|ID, %s|", yytext);

}

[0-9]+[A-Za-z0-9]\* {

printf("|ELEMENT, %s|", yytext);

}

[ \t]+ { /\* nada \*/ }

\n { putchar('\n'); }

. { /\* nada \*/ }

%%

int main(void) {

printf(">> Scanner iniciado. Escribe los comandos y pulsa Enter.\n");

yylex();

return 0;

}

Archivo en bison:

%{

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

/\* Prototipo de yylex() generado por Flex \*/

extern int yylex(void);

extern int yylineno;

extern char \*yytext;

void yyerror(const char \*s);

/\* (Opcional) Función auxiliar para liberar un char\* que está guardado

en un char\*\*, si alguna vez decides usarla. \*/

static void free\_strptr(char \*\*p) {

if (\*p) {

free(\*p);

\*p = NULL;

}

}

%}

/\* 1) Declaración de tokens (coinciden con los return del scanner.l) \*/

%token SETUNION

%token CLEARSET

%token PRINTSET

%token SHOWSETS

%token DELETESET

%token UNIONSET

%token INTERSECTION

%token SETS

%token SET

%token ASSIGN

%token LBRACE

%token RBRACE

%token COMMA

%token SEMICOLON

%token EXITCMD /\* palabra reservada para salir \*/

%token ID

%token ELEMENT

%token NEWLINE /\* opcional, si quieres contar líneas \*/

/\* 2) Unión de valores semánticos \*/

%union {

char \*str; /\* usado para ID y ELEMENT \*/

}

/\* 3) Asociar tokens que llevan valor a la unión \*/

%type <str> ID

%type <str> ELEMENT

/\* 4) Precedencia (si llegara a ser necesaria) \*/

%left COMMA

%left ASSIGN

/\* 5) Prototipo de yyparse \*/

%language "C"

%%

/\*------------------------------------------------------------

\* Gramática

\*------------------------------------------------------------\*/

program:

/\* vacío \*/

| program statement

;

statement:

instruction SEMICOLON

{

/\* Si llegamos aquí es porque instruction se procesó e imprimió en su propia acción \*/

}

| EXITCMD SEMICOLON

{

printf(">> Se recibio 'Exit'. Terminando la ejecucion del analizador.\n");

exit(0);

}

| error SEMICOLON

{

yyerrok; /\* limpiamos el estado de error \*/

printf("\*\* Error de sintaxis (línea %d). Se recupera hasta ';'.\n", yylineno);

}

;

/\* instruction: cada comando válido con sus parámetros \*/

instruction:

SETUNION ID COMMA ID

{

printf(">> Ejecutando: SetUnion con conjuntos '%s' y '%s'\n", $2, $4);

free($2);

free($4);

}

| CLEARSET ID

{

printf(">> Ejecutando: ClearSet sobre el conjunto '%s'\n", $2);

free($2);

}

| PRINTSET ID

{

printf(">> Ejecutando: PrintSet de '%s'\n", $2);

free($2);

}

| SHOWSETS

{

printf(">> Ejecutando: ShowSets (mostrar todos los conjuntos)\n");

}

| DELETESET ID

{

printf(">> Ejecutando: DeleteSet del conjunto '%s'\n", $2);

free($2);

}

| UNIONSET ID COMMA ID

{

printf(">> Ejecutando: Union (operación unión) de '%s' y '%s'\n", $2, $4);

free($2);

free($4);

}

| INTERSECTION ID COMMA ID

{

printf(">> Ejecutando: Intersection de '%s' y '%s'\n", $2, $4);

free($2);

free($4);

}

| SETS

{

printf(">> Ejecutando: Sets (definir o listar estructuras de datos de conjuntos)\n");

}

| SET ID ASSIGN LBRACE element\_list RBRACE

{

printf(">> Ejecutando: Set '%s' := { ", $2);

/\* element\_list ya imprimió todos los elementos separados por comas \*/

printf(" }\n");

free($2);

}

;

/\* element\_list: lista recursiva de elementos separados por comas \*/

element\_list:

ELEMENT

{

printf("%s", $1);

free($1);

}

| element\_list COMMA ELEMENT

{

printf(", %s", $3);

free($3);

}

;

%%

/\*------------------------------------------------------------

\* Código en C adicional

\*------------------------------------------------------------\*/

void yyerror(const char \*s) {

fprintf(stderr, "\*\* yyerror (línea %d): %s\n", yylineno, s);

}

int main(void) {

printf(">> Analizador sintactico iniciado. Escribe tus comandos, terminados con ';'.\n");

printf(" Para salir, escribe: Exit;\n\n");

yylineno = 1;

yyparse();

return 0;

}

El analizador léxico detecta la cadena de símbolos ingresada y la compara con el analizador sintáctico para detectar que la cadena de tokens sea válida. Cuando el analizador semántico verifica que la cadena de tokens es válida, regresa la descripción del comando y la imprime en pantalla.